# Mục tiêu:

* Hiểu và sử dụng được các ngoại vi UART, I2C, SPI
* Hiểu cách giao tiếp với RTC, EEPROM

# Tham khảo:

* Tài liệu hướng dẫn thí nghiệm, chương 7, 9, 11
* Atmel-2505-Setup-and-Use-of-AVR-Timers\_ApplicationNote\_AVR130.pdf

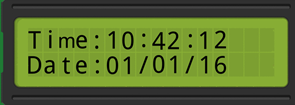
# Bài 1

1. Kết nối chân TxD và RxD của UART0 vào vào tín hiệu UART\_TxD0 và UART\_RxD0 trên header J85 ở khối UART.
2. Kết nối dây USB-Serial vào kit thí nghiệm
3. Setup chương trình Hercules với baudrate 9600, 8 bit data, no parity, 1 stop, no handshake.
4. Sử dụng các ví dụ mẫu trong tài liệu thí nghiệm, viết chương trình khởi động UART0 với các thông số như trên, chờ nhận một byte từ UART0 và phát ngược lại UART0.
5. Dùng Hercules truyền một ký tự xuống kit và quan sát các dữ liệu nhận được để kiểm tra hoạt động chương trình.

(Lưu ý: tần số xung clock cho CPU trên kit thí nghiệm là 8Mhz)

# Bài 2

1. Kết nối các tín hiệu SDA và SCL của AVR vào các tín hiệu tương ứng trên module RTC. Kết nối 1 chân port vào tín hiệu MFP. Kết nối LCD 16x2 vào 1 port của AVR
2. Viết chương trình con khởi động RTC với thời gian hiện hành, cấu hình xung MFP tần số 1Hz. Sau đó cứ mỗi cạnh lên của MFP, đọc các giá trị ngày tháng năm giờ phút giây của RTC và cập nhật lên LCD



1. Biên dịch chương trình và quan sát LCD để kiểm tra chương trình.

# Bài 3

1. Kết nối các tín hiệu MOSI, SCK của port SPI từ AVR đến tín hiệu SDI và CLK của khối thanh ghi dịch. Kết nối 2 chân port khác vào tín hiệu nCLR và LATCH. Kết nối ngõ ra của thanh ghi dịch vào Bar LED
2. Kết nối các tín hiệu UART như ở bài 1.
3. Viết chương trình nhận 1 giá trị từ UART và xuất ra Bar Led sử dụng SPI.

# Bài 4

1. Kết nối các tín hiệu MOSI, MISO, SCK của port SPI từ AVR các tín hiệu tương ứng trên header J80. Kết nối 1 chân port khác vào tín hiệu nCS.
2. Kết nối các tín hiệu UART như ở bài 1.
3. Kết nối 1 port vào Bar LED.
4. Viết chương trình đếm số ký tự nhận được từ UART và xuất ra Bar Led, cứ mỗi lần có 1 byte nhận được, số đếm tăng lên 1 và được ghi vào EEPROM. Khi vi xử lý mất điện và có lại, số đếm được đọc ra từ EEPROM và lấy làm giá trị bắt đầu.

# Bài 5

1. Kết nối các tín hiệu UART như ở bài 1.
2. Kết nối 1 port vào Bar LED.
3. Viết chương trình đếm số ký tự nhận được từ UART và xuất ra Bar Led, cứ mỗi lần có 1 byte nhận được, số đếm tăng lên 1 và được ghi vào EEPROM nội của AVR. Khi vi xử lý mất điện và có lại, số đếm được đọc ra từ EEPROM nội và lấy làm giá trị bắt đầu.

Nhóm L09

Tổ 3

Trịnh Quang Huy 2110211

Nguyễn Ngọc Khanh 2111474

Trương Quốc Khánh 2111501

# Bài 1

1. Trả lời các câu hỏi
2. Với tần số là 8Mhz, baudrate thực tế sẽ sai lệch với mong muốn là 9600 như thế nào?

* Ta có tốc độ baud = 2U2X0. với fosc = 8MHz và U2X0 = 0 (không nhân đôi tốc độ) thì ta tính ta được giá trị số thập phân phải nạp cho UBRR0 để đạt được 9600 là giá trị 613/12 , xấp xỉ 51,0833.
* Khi nạp UBRR0, ta phải nạp giá trị số nguyên nên sẽ có xảy ra sai số giữa số chính xác và số nguyên được làm tròn
* Tốc độ baud chính xác ở giá trị UBRR0 = 51 là

tốc độ baud = 2U2X0.= = 9615,3846

* Vậy sai số là : = 0,1603%

1. Cờ UDRE dùng để làm gì?

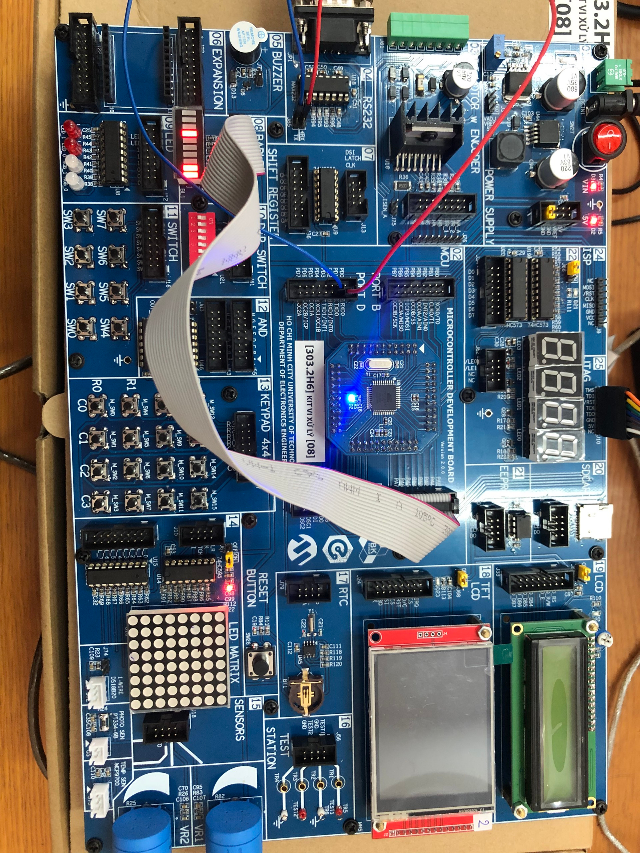
* Cờ UDRE (Data Register Empty Flag) là một trong các cờ trạng thái của UART trong vi điều khiển, được sử dụng để kiểm tra xem thanh ghi UDR (UART Data Register) đã sẵn sàng để gửi dữ liệu mới hay chưa.
* Khi một byte dữ liệu được gửi đi, UDR sẽ trống và cờ UDRE sẽ được set lên để thông báo cho vi điều khiển rằng nó có thể gửi dữ liệu mới vào UDR để truyền đi. Việc sử dụng cờ UDRE giúp tránh tình trạng gửi dữ liệu trước khi UDR đã sẵn sàng và tránh mất dữ liệu trong quá trình truyền.

1. Sự khác nhau giữa hardware UART và software UART (bit-banging UART)

* Hardware UART và software UART (hay bit-banging UART) là hai phương pháp để giao tiếp qua UART (Universal Asynchronous Receiver/Transmitter).
* Hardware UART được tích hợp sẵn trong vi điều khiển hoặc một số chip điều khiển khác. Nó hoạt động với tốc độ cao và có độ chính xác cao hơn. Hardware UART sử dụng các phần cứng riêng biệt để xử lý dữ liệu đầu vào/đầu ra, điều khiển đồng bộ dữ liệu và đồng bộ với tốc độ baudrate đã được thiết lập. Điều này giúp giảm thiểu tối đa các lỗi trong quá trình truyền dữ liệu.
* Software UART hay bit-banging UART là một phương pháp được thực hiện thông qua phần mềm trên vi điều khiển, thường được sử dụng trong các vi điều khiển không tích hợp sẵn UART hoặc khi cần tùy chỉnh độ phức tạp của giao tiếp UART. Software UART sử dụng các chân IO của vi điều khiển để tạo tín hiệu UART và đọc các tín hiệu vào để phân tích dữ liệu. Phương pháp này đòi hỏi nhiều xử lý phần mềm hơn, do đó thường yêu cầu thời gian xử lý cao hơn và có độ chính xác thấp hơn so với hardware UART.

1. Chân TxD0 và chân RxD0 của UART0 là chân port nào?

Chân TxD0 và chân RxD0 của UART0 là chân PortD, cụ thể là chân PD1 và PD0.

1. Atmega324 có bao nhiêu phần cứng UART?

Atmega324 có 2 phần cứng UART: UART0 và UART1

1. Mã nguồn chương trình với chú thích

|  |
| --- |
| .DEF DATA\_REC=R20  .DEF DATA\_TRANS=R21  .ORG 0  RJMP MAIN  .ORG $40  MAIN:    LDI R16,HIGH(RAMEND)  OUT SPH,R16  LDI R16,LOW(RAMEND)  OUT SPL,R16  LDI R16,$FF  OUT DDRA,R16  RCALL USART\_INT  START:  RCALL USART\_REC  MOV DATA\_TRANS,DATA\_REC  MOV R19,DATA\_REC  OUT PORTA,R19  RCALL USART\_TRANS  RJMP START  USART\_INT:  LDI R16,(1<<TXEN0)|(1<<RXEN0) ;CHO PHEP BO PHAT/THU  STS UCSR0B,R16  LDI R16,(1<<UCSZ01)|(1<<UCSZ00) ;8 BIT DATA/KHONG KT CHAN LE/1 STOP BIT  STS UCSR0C,R16  LDI R16,0  STS UBRR0H,R16  LDI R16,51 ;BAUD RATE=9600 UNG VOI FOSC =8MHZ  STS UBRR0L,R16    RET  USART\_TRANS:  LDS R17,UCSR0A  SBRS R17,UDRE0 ;KIEM TRA UDR0 CO TRONG KHONG(UDRE0=1?)  RJMP USART\_TRANS  STS UDR0,DATA\_TRANS ;KHI UDR0 TRONG THI CHEP DU LIEU TU R16 VAO UDR0  RET  USART\_REC:  LDS R17,UCSR0A  SBRS R17,RXC0  RJMP USART\_REC  LDS DATA\_REC,UDR0 ;NAP DU LIEU TU UDR0 VAO DATA\_REC  RET |

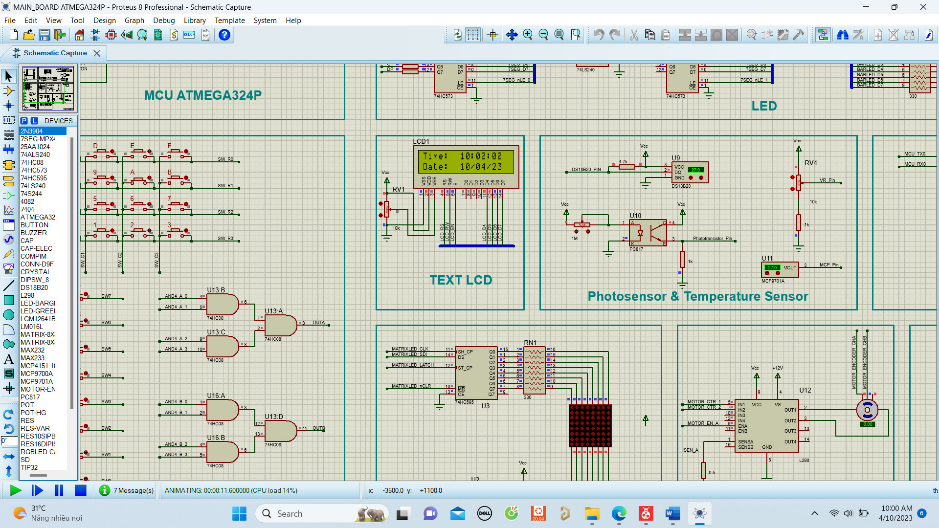
# Bài 2

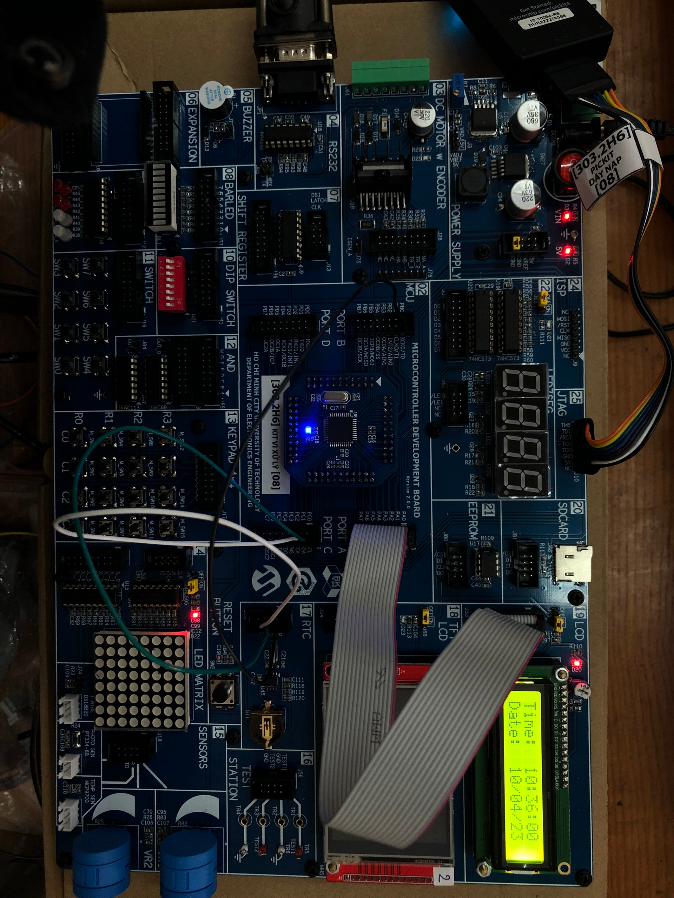
1. Trả lời các câu hỏi
   1. Các chân SCL, SDA là chân nào của AVR?

Các chân SCL (Serial Clock) và SDA (Serial Data) của AVR là:

+ SCL: chân PD0 hoặc chân PC0

+ SDA: chân PD1 hoặc chân PC1

* 1. Vẽ hình mô tả kết nối trong bài thí nghiệm



1. Mã nguồn và chú thích

|  |
| --- |
| .DEF COUNT = R20  .EQU LCD = PORTA ; PORTA HIEN THI LCD  .EQU LCD\_DR = DDRA  .EQU RS = 0 ; BIT RS PA0  .EQU RW = 1 ; BIT RW PA1  .EQU E = 2 ; BIT E PA2  .EQU SCL = 0 ; KI HIEU CHAN SCL  .EQU SDA = 1 ; KI HIEU CHAN SDA  .EQU STO = 7 ; BIT CHO PHEP OSC RTC  .EQU VBATEN = 3 ; BIT CHO PHEP NGUON DU PHONG  .EQU NULL = 0X00 ; MA KET THUC CHUOI KI TU  .EQU CTL\_BYTE = 0B11011110  .EQU RTC\_BUF = 0X200  .ORG 0  RJMP MAIN  .ORG 0X40  MAIN: LDI R16, HIGH(RAMEND)  OUT SPH, R16  LDI R16, LOW(RAMEND)  OUT SPL, R16 ; DUA STACK LEN DIA CHI CAO  LDI R16, 0XFF  OUT LCD\_DR, R16 ; PORTA LA PORT XUAT  LDI R16, 0X00  OUT LCD, R16 ; GIA TRI BAN DAU PORTA = 0  LDI R16, 250  RCALL DELAY\_US ; CTC DELAY 100US X R16 = 25MS  LDI R16, 250  RCALL DELAY\_US ; CTC DELAY 100US X R16 = 25MS  CBI LCD, RS ; RS = 0 GHI LENH  LDI R17, 0X30 ; MA LENH 30 LAN 1  RCALL OUT\_LCD4  LDI R16, 42  RCALL DELAY\_US ; DELAY 4,2MS  CBI LCD, RS ; RS = 0 GHI LENH  LDI R17, 0X30 ; MA LENH 30 LAN 2  RCALL OUT\_LCD4  LDI R16, 2  RCALL DELAY\_US ; DELAY 200US  CBI LCD, RS ; RS = 0 GHI LENH  LDI R17, 0X30 ; MA LENH 30 LAN 3  RCALL OUT\_LCD4  LDI R16, 2  RCALL DELAY\_US ; DELAY 200US  CBI LCD, RS ; RS = 0 GHI LENH  LDI R17, 0X20 ; MA LENH 20  RCALL OUT\_LCD4  LDI R18, 0X28 ; FUNCTION SET 2 DONG FONT 5X8 MODE 4 BIT  LDI R19, 0X01 ; CLEAR DISPLAY  LDI R20, 0X0C ; DISPLAY ON, CON TRO OFF  LDI R21, 0X06 ; ENTRY MODE SET DICH PHAI CON TRO, DDRAM TANG 1 DIA CHI  RCALL INIT\_LCD4 ; CTC KHOI DONG LCD 4 BIT  RCALL TWI\_INIT  START:  LDI R16, 1 ; CHO 100US  RCALL DELAY\_US  CBI LCD, RS ; RS = 0 GHI LENH  LDI R17, 0X01 ; XOA MAN HINH  RCALL OUT\_LCD  LDI R16, 20  RCALL DELAY\_US ; DOI 2MS SAU LENH CLEAR DISPLAY  LDI R17, 0X80 ; CON TRO BAT DAU O DONG 1  RCALL CURS\_POS ; XUAT LENH RA LCD  LDI ZH, HIGH(MSG1<<1)  LDI ZL, LOW(MSG1<<1) ; Z TRO DAU BANG MSG1  RCALL MSG\_DISP ; GHI MSG1 RA LCD  LDI R17, 0XC0 ; CON TRO BAT DAU O DONG 2  RCALL CURS\_POS ; XUAT LENH RA LCD  LDI ZH, HIGH(MSG2<<1)  LDI ZL, LOW(MSG2<<1) ; Z TRO DAU BANG MSG2  RCALL MSG\_DISP ; GHI MSG2 RA LCD  ;---------------------------------------------------------------------------------  ; DAT BIT SQWEN = 1 , RS[2:0] = 000 CHO DAO DONG 1HZ  ; XUAT RA CHAN MFP  ;---------------------------------------------------------------------------------  RCALL TWI\_START ; PHAT XUNG START  LDI R17, (CTL\_BYTE | 0X00) ; TRUY XUAT GHI RTC\_TCCR  RCALL TWI\_WRITE ; GHI RTC + W  LDI R17, 0X07 ; DIA CHI THANH GHI CONTROL  RCALL TWI\_WRITE  LDI R17, 0B01000000 ; MFP XUAT XUNG 1HZ  RCALL TWI\_WRITE  RCALL TWI\_STOP  ;-----------------------------------------------------------------------------------  ; DOC CAC THANH GHI 0X00 DEN 0X06 RTC  ;-----------------------------------------------------------------------------------  START1: LDI XH, HIGH(RTC\_BUF)  LDI XL, LOW(RTC\_BUF)  LDI COUNT, 7  RCALL TWI\_START ; PHAT XUNG START  LDI R17, (CTL\_BYTE | 0X00) ; TRUY XUAT GHI RTC\_TCCR  RCALL TWI\_WRITE ; GHI RTC + W  LDI R17, 0X00 ; DIA CHI THANH GHI 0X00  RCALL TWI\_WRITE  RCALL TWI\_START  LDI R17, (CTL\_BYTE | 0X01) ; TRUY XUAT DOC RTC\_TCCR  RCALL TWI\_WRITE ; GHI RTC + R  RTC\_RD: RCALL TWI\_READ  ST X+, R17  DEC COUNT  BRNE RTC\_RD  RCALL TWI\_NAK  RCALL TWI\_STOP  ;-----------------------------------------------  CAIDATRTC:  LDI R22, 0X36  LDI XH, HIGH(RTC\_BUF)  LDI XL, LOW(RTC\_BUF)  ST X, R22  LDI R22, 0X10  LDI XH, HIGH(RTC\_BUF+1)  LDI XL, LOW(RTC\_BUF+1)  ST X, R22  LDI R22, 0b01010000  LDI XH, HIGH(RTC\_BUF+2)  LDI XL, LOW(RTC\_BUF+2)  ST X, R22  LDI R22, 0X10  LDI XH, HIGH(RTC\_BUF+4)  LDI XL, LOW(RTC\_BUF+4)  ST X, R22    LDI R22, 0X04  LDI XH, HIGH(RTC\_BUF+5)  LDI XL, LOW(RTC\_BUF+5)  ST X, R22  LDI R22, 0X23  LDI XH, HIGH(RTC\_BUF+6)  LDI XL, LOW(RTC\_BUF+6)  ST X, R22  ;-----------------------------------------------  ; HIEN THI TIME GIO PHUT GIAY  ;-----------------------------------------------  START2: LDI R17, 0X0C ; XOA CON TRO  CBI LCD, RS  LDI R16, 1 ; CHO 100US  RCALL DELAY\_US  RCALL OUT\_LCD  LDI COUNT, 3  LDI R17, 0X87 ; CON TRO BAT DAU O VI TRI GIO DONG 1  RCALL CURS\_POS ; XUAT LENH RA LCD  DISP\_NXT1:  LDI XH, HIGH(RTC\_BUF+2)  LDI XL, LOW(RTC\_BUF+2)  LD R17, -X ; LAY DATA  CPI COUNT, 1 ; DATA = SEC  BRNE D\_NXT ; NEU KHAC 1 HIEN THI TIEP  CBR R17, (1<<STO) ; XOA BIT ST  D\_NXT: RCALL NUM\_DISP  DEC COUNT  BREQ QUIT1  LDI R17, ':'  SBI LCD, RS  LDI R16, 1  RCALL DELAY\_US  RCALL OUT\_LCD  RJMP DISP\_NXT1  ;----------------------------------------------------------------  ; HIEN THI NGAY THANG NAM  ;--------------------------------------------------------------------  QUIT1:  LDI XH, HIGH(RTC\_BUF+4) ; X TRO BUFFER RTC NGAY  LDI XL, LOW(RTC\_BUF+4)  LDI COUNT, 3  LDI R17, 0XC7 ; CON TRO BAT DAU O DONG 2 VI TRI NGAY  RCALL CURS\_POS ; XUAT LENH RA LCD  DISP\_NXT2:  LD R17, X+  RCALL NUM\_DISP  DEC COUNT  BREQ HERE  LDI R17, '/'  SBI LCD, RS  LDI R16, 1  RCALL DELAY\_US  RCALL OUT\_LCD  RJMP DISP\_NXT2  HERE: RJMP HERE  ;-----------------------------------------------------------  NUM\_DISP:  PUSH R17 ; CAT DATA  SWAP R17 ; HOAN VI 2 NIBBLE  ANDI R17, 0X0F ; CHE BCD CAO  ORI R17, 0X30 ; CHUYEN SANG MA ASCII  SBI LCD, RS  LDI R16, 1  RCALL DELAY\_US  RCALL OUT\_LCD ; HIEN THI GIA TRI  POP R17 ; PHUC HOI DATA  ANDI R17, 0X0F ; CHE BCD THAP  ORI R17, 0X30 ; CHUYEN SANG MA ASCII  SBI LCD, RS  LDI R16, 1  RCALL DELAY\_US  RCALL OUT\_LCD ; HIEN THI GIA TRI  RET  ;--------------------------------------------------------  ; MSG\_DISP HIEN THI CHUOI KI TU KET THUC BANG NULL DAT TRONG FLASH ROM  ; INPUT Z CHUA DI CHI DAU CHUOI KI TU  ; OUTPUT HIEN THI CHUOI KI TU TAI VI TRI CON TRO HIEN HANH  ; SU DUNG R16, R17 CTC DELAY\_US, OUT\_LCD  MSG\_DISP:  LPM R17, Z+ ; LAY MA ASCII KI TU TU FLASH ROM  CPI R17, NULL ; KIEM TRA KI TU KET THUC  BREQ EXIT\_MSG ; NEU LA NULL THOAT  LDI R16, 1  RCALL DELAY\_US  SBI LCD, RS ; RS = 1 GHI DATA HIEN THI LCD  RCALL OUT\_LCD  RJMP MSG\_DISP ; TIEP TUC HIEN THI KI TU  EXIT\_MSG: RET  ;---------------------------------------------------------------------  ; INIT LCD4 KHOI DONG LCD GHI 4 BYTE MA LENH THEO GIAO TIEP 4 BIT  ; FUNCTION SET R18 = 0X28 2 DONG FONT 5X8 GIAO TIEP 4 BIT  ; CLEAR DISPLAY R19 = 0X01 XOA MAN HINH  ; DISPLAY ON/OFF CONTROL R20 = 0X0C MAN HINH ON, CON TRO OFF  ; RENTRY MODE SET R21 = 0X06 DICH PHAI CON TRO , DC DDRAM TANG LEN 1 DVI  INIT\_LCD4:  CBI LCD, RS ; GHI LENH  MOV R17, R18  RCALL OUT\_LCD  MOV R17, R19  RCALL OUT\_LCD  LDI R16, 20  RCALL DELAY\_US  MOV R27, R20  RCALL OUT\_LCD  MOV R17, R21  RCALL OUT\_LCD  RET  ;--------------------------------------------------------  ; OUT LCD4 GHI MA LENH/ DATA RA LCD  ; INPUT R17 CHUA MA LENH/ DATA 4 BIT CAO  OUT\_LCD4: OUT LCD, R17  SBI LCD, E  CBI LCD, E  RET  ;------------------------------------------------------------  ; OUT\_LCD GHI 1 BYTE MA LENH/DATA RA LCD  ; CHIA LAM 2 LAN GHI 4 BIT  ; INPUT R17 CHUA MA LENH/DATA  ; RS =0/1 LENH/DATA  ; RW = 0 GHI  ; SU DUNG OUT\_LCD4  OUT\_LCD: LDI R16, 1  RCALL DELAY\_US  IN R16, LCD ; DOC PORT LCD  ANDI R16, (1<<RS) ; LOC BIT RS  PUSH R16  PUSH R17  ANDI R17, 0XF0 ; LAY 4 BIT CAO  OR R17, R16 ; GHEP BIT RS  RCALL OUT\_LCD4 ; GHI RA LCD  LDI R16, 1  RCALL DELAY\_US  POP R17  POP R16  SWAP R17  ANDI R17, 0XF0 ; LAY 4 BIT THAP CHUYEN THANH CAO  OR R17, R16 ; GHEP BIT RS  RCALL OUT\_LCD4 ; GHI RA LCD  RET  ;------------------------------------------------------------------------------  ; CURS\_POS DAT CON TRO TAI DIA CHI CO TRONG R17  CURS\_POS:  LDI R16, 1  RCALL DELAY\_US  CBI LCD, RS ; GHI LENH  RCALL OUT\_LCD  RET  ;------------------------------------------------------------------------------  DELAY\_US:  MOV R15, R16 ; NAP DATA CHO R15  LDI R16, 200  L1: MOV R14, R16 ; NAP DATA CHO R14  L2: DEC R14  NOP  BRNE L2  DEC R15  BRNE L1  RET  ;---------------------------------------------------------  ; TWI\_INIT KHOI DONG CONG TWI  ; TOC DO TRUYEN 100K HZ  TWI\_INIT:  MOV R15, R16 ; TOC DO TRUYEN SCL = 100K HZ  STS TWBR, R17  LDI R17, 1  STS TWSR, R17 ; HE SO DAT TRUOC = 4  LDI R17, (1<<TWEN) ; CHO PHEP TWI  STS TWCR, R17  RET  ;--------------------------------------------------------  TWI\_START:  LDI R17, (1<<TWEN) | (1<<TWSTA) | (1<<TWINT) ; CHO PHEP TWI, START, XOA TWINT  STS TWCR, R17  WAIT\_STA:  LDS R17, TWCR ; DOC CO TWINT  SBRS R17, TWINT ; DOI CO TWINT = 1 BAO TRUYEN XONG  RJMP WAIT\_STA  RET  ;--------------------------------------------------------------  TWI\_WRITE:  STS TWDR, R17 ; GHI DATA  LDI R17, (1<<TWEN) | (1<<TWINT) ; CHO PHEP TWI, XOA TWINT  STS TWCR, R17  WAIT\_WR:  LDS R17, TWCR ; CHO CO TWINT = 1 BAO TRUYEN XONG  SBRS R17, TWINT  RJMP WAIT\_WR  RET  ;--------------------------------------------------------------  TWI\_READ:  LDI R17, (1<<TWEN) | (1<<TWINT) | (1<<TWEA) ; CHO PHEP TWI, XOA TWINT , TRA ACK  STS TWCR, R17  WAIT\_RD:  LDS R17, TWCR ; CHO CO TWINT = 1 BAO TRUYEN XONG  SBRS R17, TWINT  RJMP WAIT\_RD  LDS R17, TWDR ; DOC DATA THU DC  RET  ;--------------------------------------------------------------  TWI\_NAK:  LDI R17, (1<<TWEN) | (1<<TWINT) ; CHO PHEP TWI, XOA TWINT , TRA NAK  STS TWCR, R17  WAIT\_NAK:  LDS R17, TWCR ; CHO CO TWINT = 1 BAO TRUYEN XONG  SBRS R17, TWINT  RJMP WAIT\_NAK  RET  ;------------------------------------------------------------  TWI\_STOP:  LDI R17, (1<<TWEN) | (1<<TWSTO) | (1<<TWINT) ; CHO PHEP TWI, XOA TWINT , STOP  STS TWCR, R17  RET  ;---------------------------------------------------------------  .ORG 0X200  MSG1: .DB "Time:",0X00  MSG2: .DB "Date:",0X00 |

# Bài 3

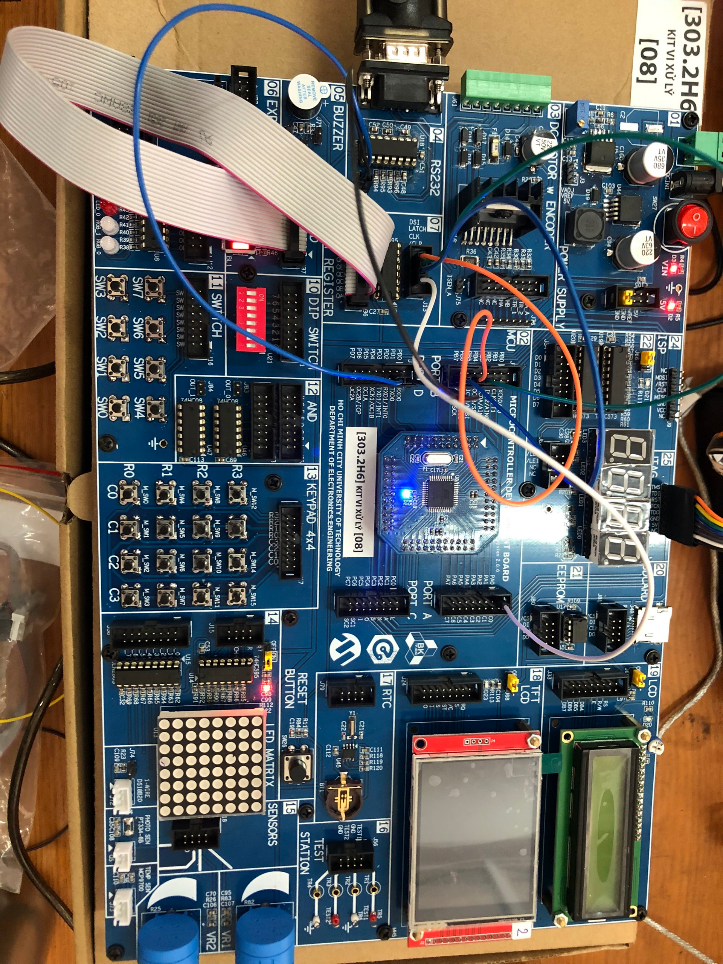
1. Trả lời các câu hỏi
   1. Theo datasheet của 74HC595, tần số cao nhất của xung nhịp đưa vào 74595 là bao nhiêu?

Theo datasheet của 74HC595, tần số cao nhất của xung nhịp đưa vào là 100 MHz.

* 1. Với clock là 8Mhz thì SPI của Atmega328 có tốc độ cao nhất là bao nhiêu?

Với clock là 8Mhz thì SPI của Atmega328 có tốc độ cao nhất là 2Mbps

1. Mã nguồn và chú thích



|  |
| --- |
| .DEF DATA\_UART\_REC=R20  .DEF DATA\_SPI\_TRANS=R21  .DEF DATA\_SPI\_REC=R22  .DEF DATA\_SHIFT=R23  .EQU CLEAR=3  .EQU SHCP=2  .EQU STCP=1  .EQU DS=0  .EQU SS=4  .EQU MOSI=5  .EQU MISO=6  .EQU SCK=7  .ORG $00  RJMP MAIN  .ORG $40  MAIN:  RCALL USART\_INT  SBI DDRA,0  SBI PORTA,0  LDI R16,(1<<SS)|(1<<SCK)|(1<<MOSI) ;KHAI BAO CAC OUTPUT SPI  OUT DDRB,R16  LDI R16,(1<<SPE0)|(1<<MSTR0)|(1<<SPR00) ;SPI MASTER  OUT SPCR0,R16  SBI PORTB,SS ;DUNG TRUYEN SPI  START:  RCALL USART\_REC  MOV DATA\_SPI\_TRANS,DATA\_UART\_REC  RCALL SPI\_TRANS\_REC  CBI PORTB,SS  SBI PORTB,SS    RJMP START  ;---------------  USART\_INT:  LDI R16,(1<<TXEN0|1<<RXEN0) ;CHO PHEP BO THU/PHAT  STS UCSR0B,R16  LDI R16,(1<<UCSZ01)|(1<<UCSZ00) ;8 BIT DATA/KHONG KT CHAN LE/1 STOP BIT  STS UCSR0C,R16  LDI R16,0  STS UBRR0H,R16  LDI R16,51 ;BAUD RATE=9600 UNG VOI FOSC =8MHZ  STS UBRR0L,R16    RET  USART\_REC:  LDS R17,UCSR0A  SBRS R17,RXC0  RJMP USART\_REC  LDS DATA\_UART\_REC,UDR0 ;NAP DU LIEU TU UDR0 VAO R16  RET  SPI\_TRANS\_REC:  ;MOV DATA\_SPI\_TRANS,DATA\_UART\_REC  OUT SPDR0,DATA\_SPI\_TRANS ;GHI DARA RA SPI  WAIT\_SPI:  IN R16,SPSR0 ;DOC CO SPIF0  SBRS R16,SPIF0 ;CO SPIEF0=1 TRUYEN SPI XONG  RJMP WAIT\_SPI ;CHO CO SPIF0=1  //IN DATA\_SPI\_REC,SPDR0 ;DOC DATA TU SPI  RET |

# Bài 4

1. Trả lời các câu hỏi
   1. Dung lượng của EEPROM 25AA1024 là bao nhiêu?

EEPROM 25AA1024 có dung lượng là 128 KB

* 1. Theo datasheet, tần số nhanh nhất của xung CK đưa vào EEPROM này là bao nhiêu?

Theo datasheet của EEPROM 25AA1024, tần số nhanh nhất của xung CK đưa vào EEPROM là 20 MHz.

1. Mã nguồn và chú thích

|  |
| --- |
|  |

# Bài 5

1. Trả lời các câu hỏi
2. Atmega324PA có dung lượng EEPROM là bao nhiêu?

Atmega324PA có dung lượng EEPROM là 4KB.

1. Liệt kê sự khác nhau giữa SRAM và EEPROM

* SRAM là loại bộ nhớ volatile (mất dữ liệu khi không còn điện), trong khi EEPROM là loại bộ nhớ non-volatile (dữ liệu được lưu giữ ngay cả khi không còn điện).
* SRAM có tốc độ truy cập nhanh hơn so với EEPROM. Thời gian truy cập của SRAM chỉ trong vài ns, trong khi EEPROM cần từ vài μs đến vài ms để truy cập.
* SRAM thường có dung lượng nhỏ hơn so với EEPROM, vì SRAM được sử dụng chủ yếu cho các mục đích lưu trữ tạm thời trong khi EEPROM được sử dụng cho lưu trữ dữ liệu lâu dài.
* EEPROM có thể được lập trình lại nhiều lần, trong khi SRAM không thể lập trình lại.
* SRAM được sử dụng chủ yếu cho các mục đích lưu trữ tạm thời, như bộ đệm hoặc bộ nhớ đệm, trong khi EEPROM được sử dụng cho các mục đích lưu trữ dữ liệu quan trọng, như các thiết lập hệ thống hoặc dữ liệu cấu hình.

1. Liệt kê sự khác nhau giữa Flash và EEPROM

* Flash chỉ là một loại EEPROM
* Flash sử dụng bộ nhớ kiểu NAND trong khi EEPROM sử dụng kiểu NOR
* Flash có khả năng xóa được
* Flash được viết lại liên tục trong khi các EEPROM khác ít khi được viết lại
* Ta dung Flash là khi cần một lượng lớn dung lượng, ở Atmega324P Flash có dung lượng 32KB trong khi EEPROM được sử dụng khi chỉ cần một lượng nhỏ dung lượng (4KB)

1. Mã nguồn chương trình với chú thích

|  |
| --- |
|  |